ISNEL Maxime L3 INFO
MARIN Estelle

# COMPTE RENDU APNEE2

Le but de cette apnée est de gérer une base de données. La base de données fournies comportait la carte des vins, avec leur nom leur degrés et leur qualité, et une liste d'achat, comportant le nom des vins et le nombres de bouteilles achetées. Elle était sous forme de fichiers texte donc les différents attributs était séparés par des tabulations. Nous devions faire des jointures entre ces deux tables de données. Un algorithme dit naïf nous a été donné et nous devions le comprendre et évaluer sa complexité. Ensuite nous avons du compléter un code dont l'algorithme reposé sur l'utilisation d'une table de hachage : contrairement à l'algorithme naïf, ce nouvel algorithme ne parcoure le ficher de la carte des vins qu'une seul fois et stocke les informations dans la table de hachage. Nos observations et les réponses aux questions posées se trouvent ci-dessous.

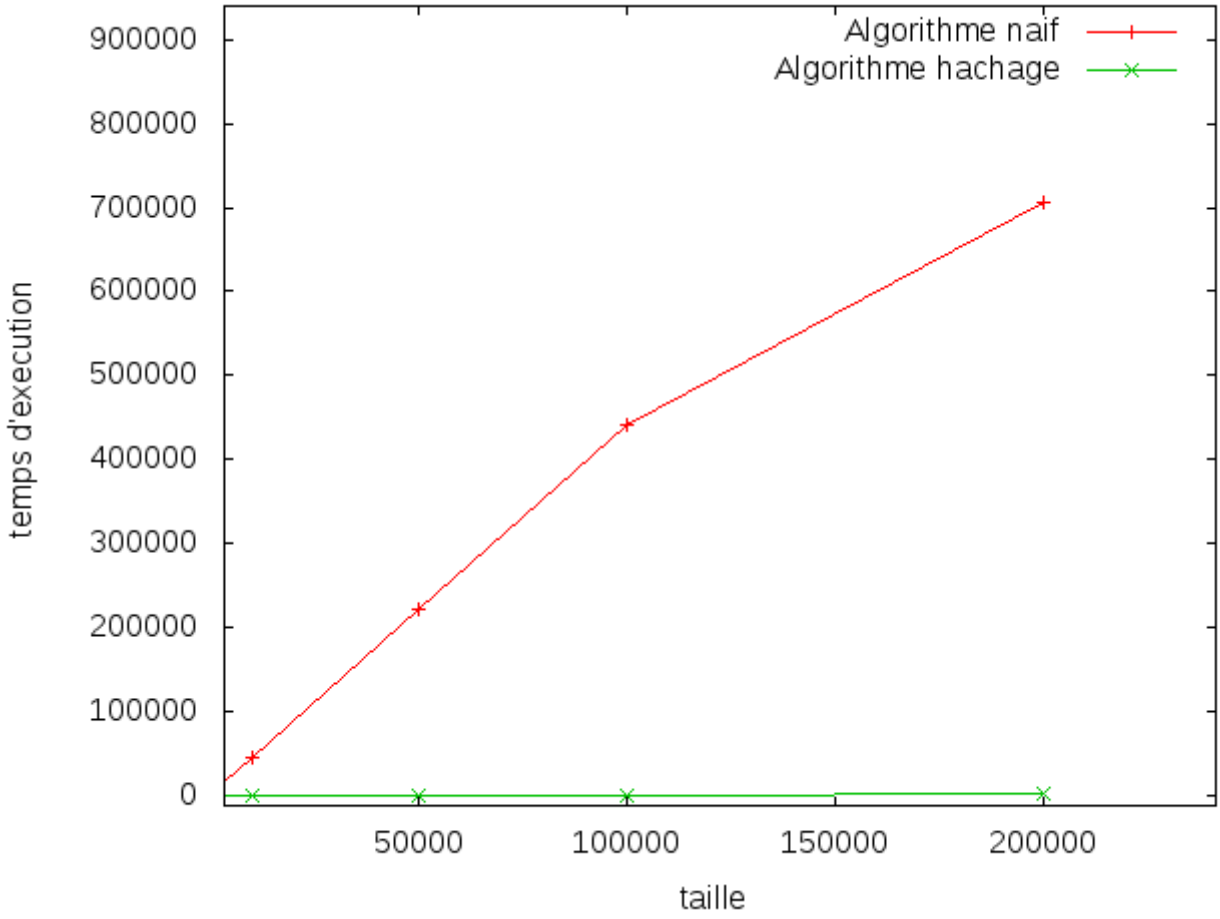
*Exercice 2 :*

```
NestedJoin() {

{*f1 premier fichier, liste d'achats*}
{*f2 deuxième fichier, carte des vins*}
Tant que !fin(f1){

        ID ← nom du vin dans f1

        Tant que ( fin(f2) et que ( nom du vin dans f2 <> ID ) ) {

                On passe a la ligne suivante
        } {* fin de la boucle de parcours du second fichier*}
        si ( nom du vin dans f2 <> ID ) alors {
                On écrit toutes les informations (nom du vin, degrés, qualité, nombre d'achats).
        }

        on repart au début du second fichiers

} {* fin de la boucle de parcours du premier fichier *}
```

La complexité de cet algorithme pire est :

nblignes(fichier1) x (nblignes(fichier2)+1)

*Exercice 4*




*Graphique du temps d'exécution des deux programmes en fonctions de la taille des listes d'achats*

On remarque que le temps d'exécution pour l'algorithme naïf est bien plus important que celui de l'algorithme de hachage. De plus, ce temps augmente considérablement avec la taille pour le premier tandis que pour le deuxième, la variation est très petite.

*Exercice 5 :*